
aimfast Documentation

Release 1.3.4

Athanaseus Ramaila

Aug 29, 2023

CONTENTS

1	Contents:	3
1.1	aimfast	3
1.2	Introduction	3
1.3	Fidelity Matrices	3
1.4	Installation	4
1.5	aimfast	8
1.6	License	18
1.7	Contribute	18
1.8	Contact us	18
2	Indices and tables	19
	Python Module Index	21
	Index	23

An Astronomical Image Fidelity Assessment Tool

CONTENTS:

1.1 aimfast

An Astronomical Image Fidelity Assessment Tool

1.2 Introduction

Image fidelity is a measure of the accuracy of the reconstructed sky brightness distribution. A related metric, dynamic range, is a measure of the degree to which imaging artifacts around strong sources are suppressed, which in turn implies a higher fidelity of the on-source reconstruction. Moreover, the choice of image reconstruction algorithm and source finder also affects the estimate on-source brightness distribution.

1.3 Fidelity Matrices

1.3.1 Image dynamic range

Dynamic range is a measure of the degree to which imaging artifacts around strong sources are suppressed, which in turn implies a higher fidelity of the on-source reconstruction. Here we determine it in three ways: Obtaining the quotient of - highest peak flux ($flux_{peak}$) and the absolute of the minimum flux ($flux_{min}$) around the peak in the residual image. - highest peak flux ($flux_{peak}$) and the rms flux ($flux_{local,rms}$) around the peak in the residual image. - highest peak flux ($flux_{peak}$) and the rms flux ($flux_{global,rms}$) in the residual image.

$$DR = \frac{flux_{peak}}{|flux_{min}|} (1) DR = \frac{flux_{peak}}{|flux_{local,rms}|} (2) DR = \frac{flux_{peak}}{|flux_{global,rms}|} (3) \quad (1.1)$$

1.3.2 Statistical moments of distribution

The mean and the variance provide information on the location (general value of the residual flux) and variability (spread, dispersion) of a set of numbers, and by doing so, provide some information on the appearance of the distribution of residual flux in the residual image. The mean and variance are calculated as follows respectively

$$MEAN = \frac{1}{n} \sum_{i=1}^n (x_i) (4) \quad (1.2)$$

and

$$VARIANCE = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 (5) \quad (1.3)$$

whereby

$$STD_DEV = \sqrt{VARIANCE}(6) \quad (1.4)$$

The third and fourth moments are the skewness and kurtosis respectively. The skewness is the measure of the symmetry of the shape and kurtosis is a measure of the flatness or peakness of a distribution. These moments are used to characterize the residual flux after performing calibration and imaging, therefore for ungrouped data, the r-th moment is calculated as follows:

$$m_r = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^r \quad (1.5)$$

The coefficient of skewness, the 3-rd moment, is obtained by

$$SKEWNESS = \frac{m_3}{m_2^{\frac{3}{2}}} \quad (1.6)$$

If there is a long tail in the positive direction, skewness will be positive, while if there is a long tail in the negative direction, skewness will be negative.

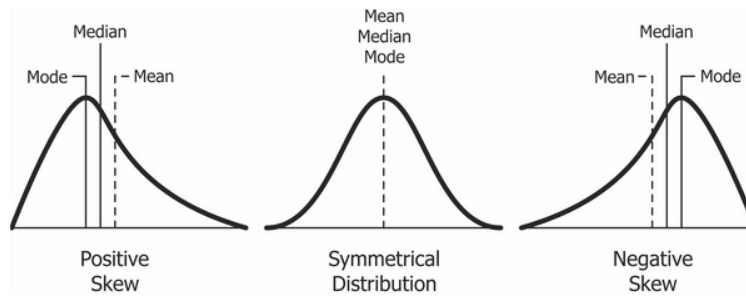


Fig. 1: Figure 1. Skewness of a distribution.

The coefficient kurtosis, the 4-th moment, is obtained by

$$KURTOSIS = \frac{m_4}{m_2^2} \quad (1.7)$$

Smaller values (in magnitude) indicate a flatter, more uniform distribution.

Furthermore, there is median absolute deviation which is a measure of how distributed is the residual data with regards to the median. This can be compared with the standard deviation to verify that the residuals are noise-like (and Gaussian).

1.4 Installation

Installation from [source](#), working directory where source is checked out

```
$ pip install .
```

This package is available on *PYPI*, allowing

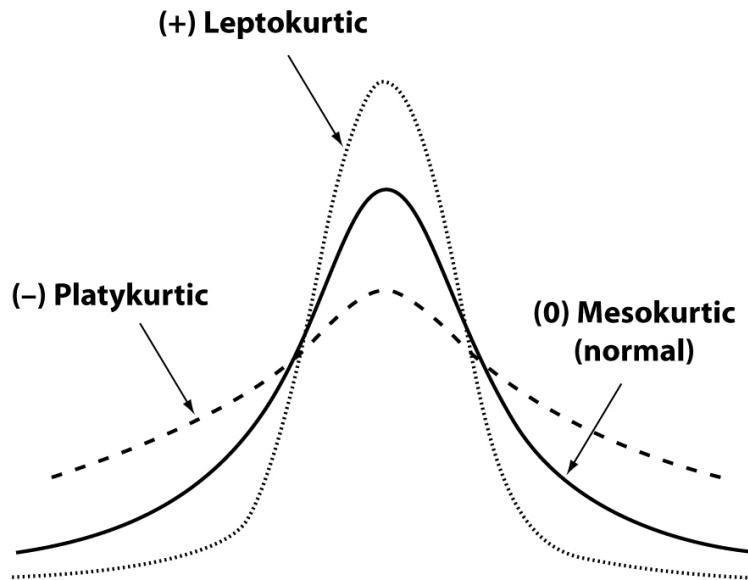


Fig. 2: Figure 2. Kurtosis of a distribution.

```
$ pip install aimfast
```

1.4.1 Command line usage

Get the statistics of the residual image

```
$ aimfast --residual-image cube.residual.fits
```

Get the residual image stats and dynamic range in one step where argument `-af` is a factor to multiply the beam area to get target peak area:

```
$ aimfast --residual-image cube.residual.fits --restored-image cube.image.fits -af 5
```

or using sky model file (e.g. `tigger.lsm.html`):

```
$ aimfast --residual-image cube.residual.fits --tigger-model model.lsm.html -af 5
```

NB: Outputs will be printed on the terminal and dumped into `fidelity_results.json` file. Moreover if the source file names are distinct the output results will be appended to the same json file.

```
$ cat fidelity_results.json
$ {"cube.residual.fits": {"SKEW": 0.124, "KURT": 3.825, "STDDev": 5.5e-05,
                          "MEAN": 4.747e-07, "MAD": 5e-05, "RMS": 5.5e-05},
  "cube.image.fits": {"DR": 35.39, "deepest_negative": 10.48,
                     "local_rms": 30.09, "global_rms": 35.39}}
```

Additionally, normality testing of the residual image can be performed using the D'Agostino (`normaltest`) and Shapiro-Wilk (`shapiro`) analysis, which returns a tuple result, e.g. `{ 'NORM': (123.3, 0.1) }`, with the z-score and p-value respectively.

```
$ aimfast --residual-image cube.residual.fits --normality-model normaltest
```

Many astronomical catalogues consist of numerous columns of source information, and often times there is an existing connection between corresponding pairs. *aimfast* allows visualising these relationships by generating interactive plots that can be easily distributed and only requires a modern web browser. A handful number of catalogue formats are supported; currently, this includes FITS, LSM and ASCII.

```
$ aimfast -catalog PictorA_1-MFS-IQUV.image.pybdsm.lsm.html -x I -y spi -x-err I_err -y-err spi_err -x-label 'Flux Density (Jy)' -y-label 'SPI' -title 'Catalog Flux vs Spectral Index' -title-size 30pt -x-size 26pt -y-size 26pt -x-maj-size 18pt -y-maj-size 18pt --html-prefix pica-hot-flux
$ aimfast -catalog PictorA_1-MFS-IQUV.image.pybdsm.lsm.html -x ra -y dec -x-err ra_err -y-err dec_err -x-label 'Right Ascension (deg)' -y-label 'Declination (deg)' -title 'Catalog RA vs DEC' -title-size 30pt -x-size 26pt -y-size 26pt -x-maj-size 18pt -y-maj-size 18pt --html-prefix pica-hot-pos
```

The results are as follows:

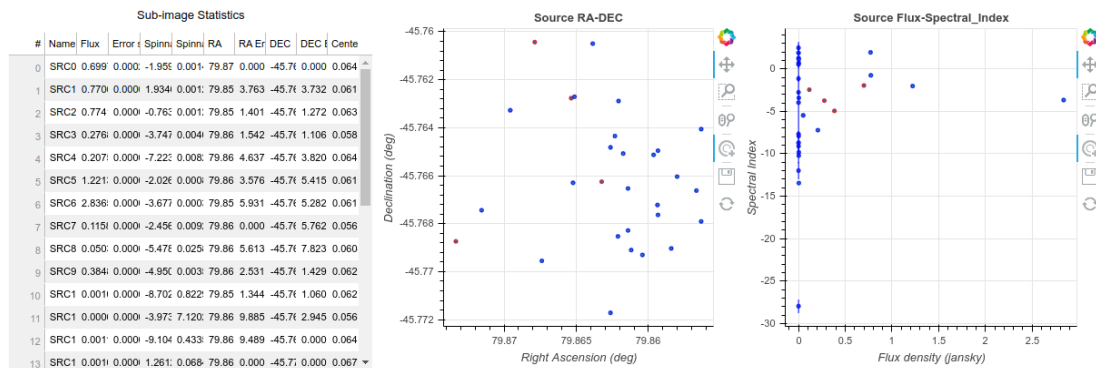


Fig. 3: Figure 3: Example of 2D plots showing source properties from a catalogue. Left: Source RA vs Dec. Right: Source flux density vs Spectral Index

Furthermore, a comparison of residual images can be performed as follows: To get random residual flux measurements in *residual1.fits* and *residual2.fits* images

```
$ aimfast --compare-residuals residual1.fits residual2.fits --area-factor 2 -dp 1590 --html-prefix cluster1 --units milli -x-size 34pt -y-size 34pt -title-size 36pt -legend-size 25pt -x-maj-size 24pt -y-maj-size 24pt -units micro
```

where `-area-factor` is the number to multiply the beam size to get area and `-dp` is the number of data points to sample. In case the beam information is missing from the image header use `-psf-image | -psf`, the point spread function file or `psf` size in arcsec, otherwise a default of 5 arcsec will be used. To get the on source residual flux measurements in a *residual1.fits* and *residual2.fits* images

```
$ aimfast --compare-residuals residual1.fits residual2.fits -catalog model.lsm.html --html-prefix cluster1 --units milli -x-size 34pt -y-size 34pt -title-size 36pt -legend-size 25pt -x-maj-size 24pt -y-maj-size 24pt -units micro
```

where `-catalog` is the catalog file name with source detections to locate exact source residuals. For random or on source residual noise comparisons, the plot on the left shows the residuals on image 1 and image 2 overlaid and the plot on the right shows the ratios. The colorbar shows the distance of the sources from the phase centre.

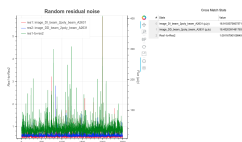


Figure 4. The random/source residual-to-residual ratio measurements

A comparison of residual images can also be performed by generating random locations on the image and calculating the statics, and overlying the results to distinguish the two images in question.

```
$ aimfast --compare-residual-subimages residual1.fits residual2.fits -cps 1004,4960,320 -
  -html-prefix cluster --units milli -x-size 15pt -y-size 15pt -bar-size 15pt
```

Generated sub-images with corresponding statistics:

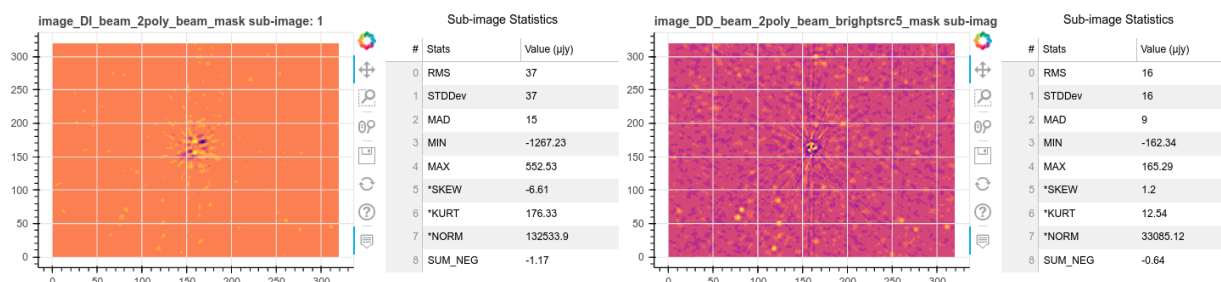


Fig. 4: Figure 5: A comparison of (320x320 Pixels) images of corrected residual data. Left: Direction-Independent corrections. Right: Direction-Dependent corrections.}

Moreover aimfast allows you to swiftly compare two (input-output) model catalogs. Currently source flux density and astrometry are examined. It returns an interactive html correlation plots, from which a .png file can be easily downloaded.

```
$ aimfast --compare-models model1.lsm.html model2.lsm.html -tol 5
```

where -tol is the tolerance to cross-match sources in arcsec. Moreover -as flag can be used to compare all source irrespective of shape (otherwise only point-like source with maj<2" are used). Access to (sumss, nvss,) online catalogs is also provided, to allow comparison of local catalogs to remote catalogs.

```
$ aimfast --compare-online model1.lsm.html --online-catalog nvss -tol 5
```

In the case where fits images are compared, aimfast can pre-install source finder of choice (pybdsf, aegean,) to generate a catalogs which are in turn compared:

```
$ aimfast --compare-images image1.fits image1.fits --source-finder pybdsf -tol 5
```

After the first run attempt one of the outputs is source_finder.yml file, which provide all the possible parameters of the source finders. Otherwise this file can be generated and edited prior to the comparison:

```
$ aimfast source-finder -gc my-source-finder.yml
$ aimfast --compare-images image1.fits image2.fits --html-prefix cluster --units milli -
  -x-size 16pt -y-size 16pt -title-size 28pt -legend-size 16pt -x-maj-size 16pt -y-maj-
  -size 16pt -bar-size 16pt -bar-major-size 14pt -units micro
```

For Flux density, the more the data points rest on the $y=x$ (or $I_{out}=I_{in}$), the more correlated the two models are.

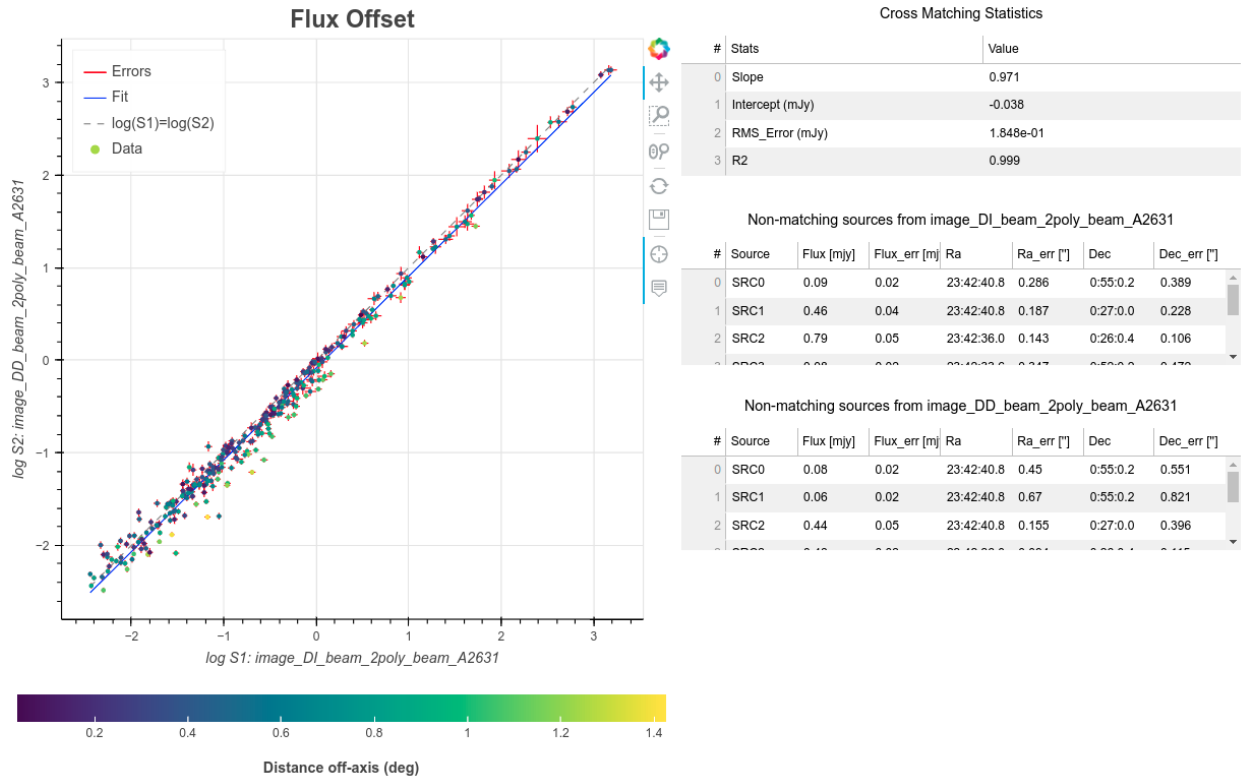


Fig. 5: Figure 6. Input-Output Flux model comparison

For astrometry, the more sources lie on the $y=0$ (Delta-position axis) in the left plot and the more points with 1 sigma (blue circle) the more accurate the output source positions.

Lastly, if you want to run any of the available source finders, generate the config file and edit then run:

```
$ aimfast source-finder -gc my-source-finder.yml
$ aimfast source-finder -c my-source-finder.yml -sf pybdsf
```

1.5 aimfast

1.5.1 aimfast modules

amifast.aimfast module

```
aimfast.aimfast.compare_models(models, tolerance=0.2, plot=True, all_sources=False, shape_limit=6.0,
                                off_axis=None, closest_only=False, prefix=None, flux_plot='log',
                                fxlabel=None, fylabel=None, ftitle=None, svg=False, title_size='16pt',
                                x_label_size='12pt', y_label_size='12pt', legend_size='10pt',
                                xmajor_size='8pt', ymajor_size='8pt', bar_size='12pt',
                                bar_major_size='8pt')
```

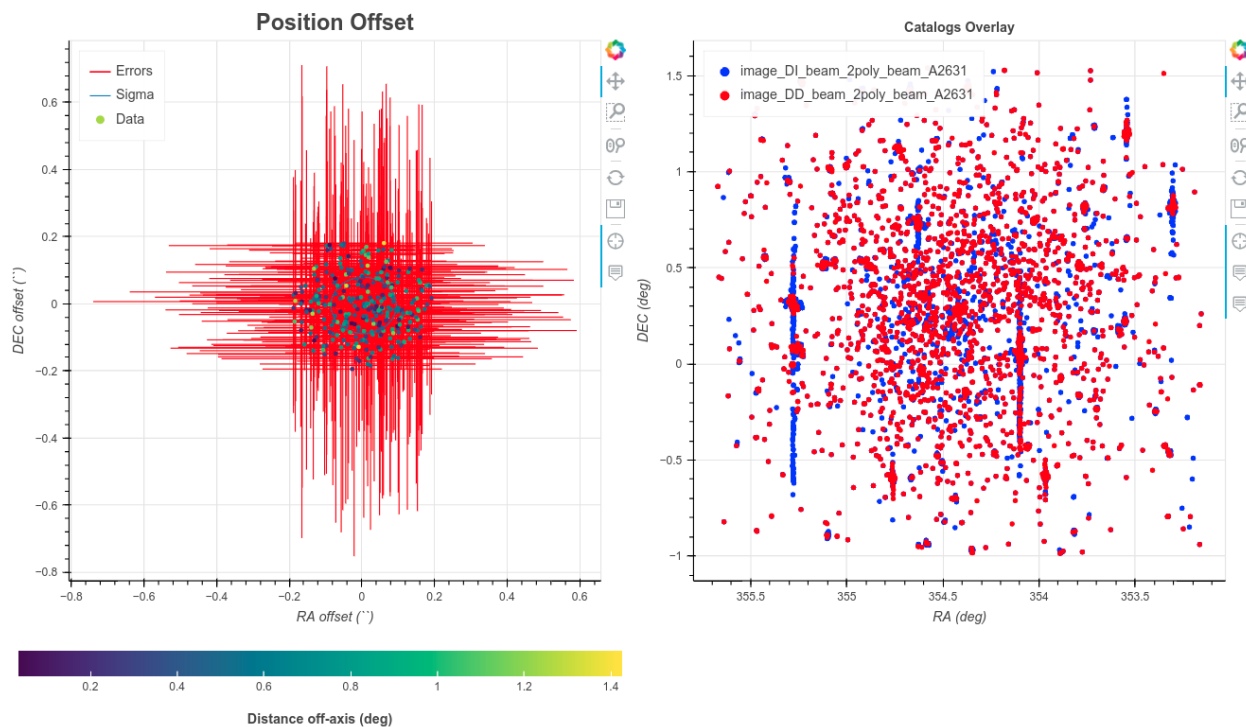


Fig. 6: Figure 7. Input-Output Astrometry model comparison

Plot model1 source properties against that of model2

Parameters

models

[dict] Tigger formatted model files e.g {model1: model2}.

tolerance

[float] Tolerance in detecting source from model 2 (in arcsec).

plot

[bool] Output html plot from which a png can be obtained.

all_source: bool

Compare all sources in the catalog (else only point-like source)

shape_limit: float

Cross match only sources with maj-axis less than this value

closest_only: bool

Returns the closest source only as the matching source

flux_plot: str

The type of output flux comparison plot (options:log,snr,inout)

prefix

[str] Prefix for output htmls

fxlabels

[str[]] X-axis labels for the flux comparison plots

fylabels

[str[]] Y-axis labels for the flux comparison plots

fylabels

[str[]] Title labels for the flux comparison plots

Returns

results

[dict] Dictionary of source properties from each model.

```
aimfast.aimfast.compare_residuals(residuals, skymodel=None, points=None, inline=False,
                                   area_factor=None, prefix=None, fov_factor=None, units='micro',
                                   title_size='14pt', xmajor_size='6pt', ymajor_size='6pt',
                                   legend_size='10pt', x_label_size='12pt', y_label_size='12pt')
```

```
aimfast.aimfast.create_logger()
```

Create a console logger

```
aimfast.aimfast.fitsInfo(fitsname=None)
```

Get fits header info.

Parameters

fitsname

[fits file] Restored image (cube)

Returns

fitsinfo

[dict] Dictionary of fits information e.g. {'wcs': wcs, 'ra': ra, 'dec': dec, 'dra': dra, 'ddec': ddec, 'raPix': raPix, 'decPix': decPix, 'b_size': beam_size, 'numPix': numPix, 'centre': centre, 'skyArea': skyArea}

```
aimfast.aimfast.fix_wcs_fits(wcs, dropaxis=2)
```

This removes the degenerated dimensions in APLpy 2.X... The input must be the object returned by aplpy.FITSFigure(). *dropaxis* is the index where to start dropping the axis (by default it assumes the 3rd,4th place).

```
aimfast.aimfast.generate_default_config(configfile)
```

Generate default config file for running source finders

```
aimfast.aimfast.get_aimfast_data(filename='fidelity_results.json', dir='.')
```

Extracts data from the json data file

```
aimfast.aimfast.get_argparser()
```

Get argument parser.

```
aimfast.aimfast.get_box(wcs, radec, w)
```

Get box of width w around source coordinates radec.

Parameters

- radec**
[tuple] RA and DEC in degrees.
- w**
[int] Width of box.
- wcs**
[atropy.wcs instance] World Coordinate System.

Returns

- box**
[tuple] A box centred at radec.

`aimfast.aimfast.get_detected_sources_properties(model_1, model_2, tolerance, shape_limit=6.0, all_sources=False, closest_only=False, off_axis=None)`

Extracts the output simulation sources properties.

Parameters

- models_1**
[file] Tigger formatted or txt model 1 file
- models_2**
[file] Compare all sources in the catalog (else only sources with maj<shape_limit)
- tolerance**
[float] Tolerace to cross-match sources
- shape_limit: float**
Cross match only sources with maj-axis less than this value
- closest_only: bool**
Returns the closest source only as the matching source
- off_axis: float**
Cross-match only sources within this distance from the centre

Returns

- (targets_flux, targets_scale, targets_position)**
[tuple] Tuple of target flux, morphology and astrometry information

`aimfast.aimfast.get_image_products(images, mask)`
Get a product of images with a mask

Parameters

images: list

List of fits images to get product

mask: str

Mask to multiply the images

Returns

prod_images: list

List of resulting fits images

`aimfast.aimfast.get_model(catalog)`

Get model object from file catalog

`aimfast.aimfast.get_region_stats(fitsname, regions_file)`

Extract flux densities measurements within the provided region

`aimfast.aimfast.get_sf_params(configfile)`

`aimfast.aimfast.get_source_overlay(sources1, sources2)`

Get source from models compare for overlay

`aimfast.aimfast.get_source_properties_from_catalog(catalog_file)`

`aimfast.aimfast.get_src_scale(source_shape)`

Get scale measure of the source in arcsec.

Parameters

source_shape

[lsm object] Source shape object from model

Returns

`(scale_out_arc_sec, scale_out_err_arc_sec)`

[tuple] Output source scale with error value

`aimfast.aimfast.image_dynamic_range(fitsname, residual, area_factor=6)`

Gets the dynamic range in a restored image.

Parameters

fitsname

[fits file] Restored image (cube).

residual

[fits file] Residual image (cube).

area_factor: int

Factor to multiply the beam area.

Returns

DR

[dict] DRs - dynamic range values.

`aimfast.aimfast.image_stats(image_data, test_normality=None, data_range=None)`

`aimfast.aimfast.json_dump(data_dict, filename='fidelity_results.json')`

Dumps the computed dictionary results into a json file.

Parameters

data_dict

[dict] Dictionary with output results to save.

filename

[str] Name of file json file where fidelity results will be dumped. Default is 'fidelity_results.json' in the current directory.

Note1

If the fidelity_results.json file exists, it will be append, and only repeated image assessments will be replaced.

`aimfast.aimfast.main()`

Main function.

`aimfast.aimfast.measure_psf(psf_file, arcsec_size=20)`

Measure point spread function after deconvolution.

Parameters

psf_file

[fits file] Point spread function file.

arcsec_size

[float] Cross section size

Returns

r0

[float] Average psf size.

`aimfast.aimfast.model_dynamic_range(lsmname, fitsname, beam_size=5, area_factor=2)`

Gets the dynamic range using model lsm and residual fits.

Parameters

fitsname

[fits file] Residual image (cube).

lsmname

[lsm.html or .txt file] Model .lsm.html from pybdsm (or .txt converted tigger file).

beam_size

[float] Average beam size in arcsec.

area_factor

[float] Factor to multiply the beam area.

Returns

DR

[dict] DRs - dynamic range values.

`aimfast.aimfast.noise_sigma(noise_image)`

Determines the noise sigma level in a dirty image with no source

Parameters

noise_image

[file] Noise image (cube).

Returns

noise_std

[float] Noise image standard deviation

`aimfast.aimfast.normality_testing(data, test_normality='normaltest', data_range=None)`

Performs a normality test on the image data.

Parameters

data

[numpy.array] Residual residual array. i.e. `fitsio.open(fitsname)[0].data`

test_normality

[str] Perform normality testing using either *shapiro* or *normaltest*.

data_range

[int] Range of data to perform normality testing.

Returns

normality

[dict] dictionary of stats props. e.g. `{ 'NORM': (123.3, 0.012) }` whereby the first element is the statistics (or average if `data_range` specified) of the datasets and second element is the p-value.

```
aimfast.aimfast.plot_aimfast_stats(fidelity_results_file, units='micro', prefix='')
```

Plot stats results if more that one residual images where assessed

```
aimfast.aimfast.plot_astrometry(models, label=None, tolerance=0.2, phase_centre=None,
                                all_sources=False, off_axis=None)
```

Plot model-model positions from lsm.html/txt models

Parameters

models

[dict] Tigger/text formatted model files e.g `{model1: model2}`.

label

[str] Use this label instead of the FITS image path when saving data.

tolerance: float

Radius around the source to be cross matched.

phase_centre

[str] Phase centre of catalog (if not already embeded)

all_source: bool

Compare all sources in the catalog (else only point-like source)

```
aimfast.aimfast.plot_model_columns(catalog_file, x, y, x_err=None, y_err=None, svg=False, x_label=None,
                                   y_label=None, title=None, html_prefix=None, title_size='16pt',
                                   x_label_size='12pt', y_label_size='12pt', legend_size='10pt',
                                   xmajor_size='6pt', ymajor_size='6pt', units='micro')
```

Plot catalog columns including their uncertainties

```
aimfast.aimfast.plot_model_data(catalog_file, html_prefix='')
```

Plotting catalog table

```
aimfast.aimfast.plot_photometry(models, label=None, tolerance=0.2, phase_centre=None,
                                 all_sources=False, flux_plot='log', off_axis=None, shape_limit=6.0)
```

Plot model-model fluxes from lsm.html/txt models

Parameters

models

[dict] Tigger/text formatted model files e.g {model1: model2}.

label

[str] Use this label instead of the FITS image path when saving data.

tolerance: float

Radius around the source to be cross matched (in arcsec).

phase_centre

[str] Phase centre of catalog (if not already embeded)

all_source: bool

Compare all sources in the catalog (else only point-like source)

```
aimfast.aimfast.plot_residuals_noise(res_noise_images, skymodel=None, label=None, area_factor=2.0,
                                     points=100)
```

Plot residual-residual or noise data

Parameters

res_noise_images: dict

Dictionary of residual images to plot {res1.fits: res2.fits}.

skymodel: file

Skymodel file to locate on source residuals (lsm.html/txt)

label

[str] Use this label instead of the FITS image path when saving data.

area_factor

[float] Factor to multiply the beam area.

points: int

Number of data point to generate in case of random residuals.

```
aimfast.aimfast.plot_subimage_stats(fitsnames, centre_coords, sizes, htmlprefix='default', title_size='12pt',
                                    x_label_size='10pt', y_label_size='10pt', bar_label_size='15pt',
                                    units='micro', svg=False)
```

Plot subimages and stats

```
aimfast.aimfast.residual_image_stats(fitsname, test_normality=None, data_range=None, threshold=None,
                                     chans=None, mask=None)
```

Gets statistical properties of a residual image.

Parameters

fitsname

[file] Residual image (cube).

test_normality

[str] Perform normality testing using either *shapiro* or *normaltest*.

data_range

[int, optional] Range of data to perform normality testing.

threshold

[float, optional] Cut-off threshold to select channels in a cube

chans

[str, optional] Channels to compute stats (e.g. 1;0~50;100~200)

mask

[file] Fits mask to get stats in image

Returns

props

[dict] Dictionary of stats properties. e.g. {'MEAN': 0.0, 'STDDDev': 0.1, 'RMS': 0.1, 'SKEW': 0.2, 'KURT': 0.3, 'MAD': 0.4, 'MAX': 0.7, 'SUM_NEG': -0.1}

Notes

If normality_test=True, dictionary of stats props becomes e.g. {'MEAN': 0.0, 'STDDDev': 0.1, 'SKEW': 0.2, 'KURT': 0.3, 'MAD': 0.4, 'RMS': 0.5, 'SUM_NEG': -0.1, 'MAX': 0.7, 'NORM': (123.3,0.012)} whereby the first element is the statistics (or average if data_range specified) of the datasets and second element is the p-value.

`aimfast.aimfast.source_finding(sf_params, sf=None)`

`aimfast.aimfast.targets_not_matching(sources1, sources2, matched_names, flux_units='milli')`

Plot model-model fluxes from lsm.html/txt models

Parameters

sources1: list

List of sources from model 1

sources2: list

List of sources Sources from model 2

matched_names: dict

Dict of names from model 2 that matched that of model 1

flux_units: str

Units of flux density for tabulated values

Returns

target_no_match1: dict

Sources from model 1 that have no match in model 2

target_no_match2: dict

Sources from model 2 that have no match in model 1

amifast.tests.test_aimfast module

1.6 License

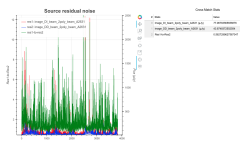
This project is licensed under the GNU General Public License v3.0 - see [license](#) for details.

1.7 Contribute

Contributions are always welcome! Please ensure that you adhere to our coding standards [pep8](#).

1.8 Contact us

Athanaseus Ramaila (aramaila@ska.ac.za)



INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

a

`aimfast.aimfast`, 8

INDEX

A

`aimfast.aimfast`
 module, 8

C

`compare_models()` (in module `aimfast.aimfast`), 8
`compare_residuals()` (in module `aimfast.aimfast`), 10
`create_logger()` (in module `aimfast.aimfast`), 10

F

`fitsInfo()` (in module `aimfast.aimfast`), 10
`fix_wcs_fits()` (in module `aimfast.aimfast`), 10

G

`generate_default_config()` (in module `aimfast.aimfast`), 10
`get_aimfast_data()` (in module `aimfast.aimfast`), 10
`get_argparser()` (in module `aimfast.aimfast`), 10
`get_box()` (in module `aimfast.aimfast`), 10
`get_detected_sources_properties()` (in module `aimfast.aimfast`), 11
`get_image_products()` (in module `aimfast.aimfast`), 11
`get_model()` (in module `aimfast.aimfast`), 12
`get_region_stats()` (in module `aimfast.aimfast`), 12
`get_sf_params()` (in module `aimfast.aimfast`), 12
`get_source_overlay()` (in module `aimfast.aimfast`), 12
`get_source_properties_from_catalog()` (in module `aimfast.aimfast`), 12
`get_src_scale()` (in module `aimfast.aimfast`), 12

I

`image_dynamic_range()` (in module `aimfast.aimfast`), 12
`image_stats()` (in module `aimfast.aimfast`), 13

J

`json_dump()` (in module `aimfast.aimfast`), 13

M

`main()` (in module `aimfast.aimfast`), 13
`measure_psf()` (in module `aimfast.aimfast`), 13

`model_dynamic_range()` (in module `aimfast.aimfast`), 14

module
 `aimfast.aimfast`, 8

N

`noise_sigma()` (in module `aimfast.aimfast`), 14
`normality_testing()` (in module `aimfast.aimfast`), 14

P

`plot_aimfast_stats()` (in module `aimfast.aimfast`), 15
`plot_astrometry()` (in module `aimfast.aimfast`), 15
`plot_model_columns()` (in module `aimfast.aimfast`), 15
`plot_model_data()` (in module `aimfast.aimfast`), 15
`plot_photometry()` (in module `aimfast.aimfast`), 15
`plot_residuals_noise()` (in module `aimfast.aimfast`), 16
`plot_subimage_stats()` (in module `aimfast.aimfast`), 16

R

`residual_image_stats()` (in module `aimfast.aimfast`), 16

S

`source_finding()` (in module `aimfast.aimfast`), 17

T

`targets_not_matching()` (in module `aimfast.aimfast`), 17